UNIX-Based Data Management System for the Mobile Satellite Propagation Experiment (PiFEx)

Anil V. Kantak

September 15, 1987

National Aeronautics and Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

JPL Publication 87-24

UNIX-Based Data Management System for the Mobile Satellite Propagation Experiment (PiFEx)

Anil V. Kantak

September 15, 1987

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

ABSTRACT

A new method is presented for handling data resulting from Mobile Satellite propagation experiments such as the Pilot Field Experiment (PiFEx) conducted by JPL. This method uses the UNIX operating system and C programming language. The data management system is implemented on a VAX minicomputer. The system automatically divides the large data file housing data from various experiments under a predetermined format into various individual files containing data from each experiment. The system also has a number of programs written in C and FORTRAN languages to allow the researcher to obtain meaningful quantities from the data at hand.

CONTENTS

Figures

Tables

SECTION 1

INTRODUCTION

The Pilot Field Experiment (PiFEx) was an attempt to mimic the proposed Mobile Satellite (MSAT) scenario. Conducted at the National Oceanic and Atmospheric Administration (NOAA) 1000-foot tower located at Denver, Colorado, PiFEx consisted of a fixed transmitting station that transmitted a known data pattern under the 8-phase modulation scheme. The transmitted signal was received by a transponder located at the top of the NOAA tower and, after frequency translation, was retransmitted to a receiver located in a van. The van containing the receiver was driven around the tower in a predetermined fashion so as to simulate the mobile receiver in the MSAT project. While being driven around the tower, the van received the signal, appropriately processed it, and finally stored it in a mass storage device for post-experiment processing.

The major aims in the PiFEx propagation experiment were to define the mobile communications channels and to test the workability of a number of new concepts used to design various components of the receiver system. These components included subsystems like the JPL-designed and -developed mechanically steered antenna for tracking and locking onto the incoming information-bearing signal, and the data acquisition and collection subsystem. The result of such an experiment is usually large amounts of data stored in either magnetic tapes or microcomputer floppy diskettes. In general, because there are such large amounts of

data, they usually reside on a minicomputer rather than a microcomputer.

One of the researcher's main tasks thus becomes accessing the data according to his needs but with a minimum of effort. Even though most minicomputer and mainframe computer systems have their own data base management systems, most of these systems have been created for commercial (e.g., banking) data and either are unsuitable for scientific computational efforts or entail a substantial investment that may not be justifiable. In general, tailoring such a commercial data-handling system to handle scientific data such as the data from the PiFEx propagation experiment would require many hours of effort and result only in extremely inefficient use of the data base management system's capabilities.

The PiFEx data are currently stored on a VAX 780 minicomputer running on the UNIX operating system. The system also has C and FORTRAN compilers. What follows in this report is a method to provide a system to manage the PiFEx data on the VAX in an interactive way. The system provides not only the file handling necessary to retrieve the desired data, but also several FORTRAN programs to generate some standard results pertaining to propagation data.

SECTION 2

DATA FORMAT

As was described before, the PiFEx experiment used a fixed transmitter to transmit a known data pattern to the transponder located at the top of the NOAA tower. The receiver was housed in a mobile van which was driven around the tower in a predetermined fashion to generate the data the researchers needed. The received signal was appropriately sampled and stored into a mass storage device by the data collection system. PiFEx hosted quite a few experiments, but the data can be categorized into four distinct classes of data:

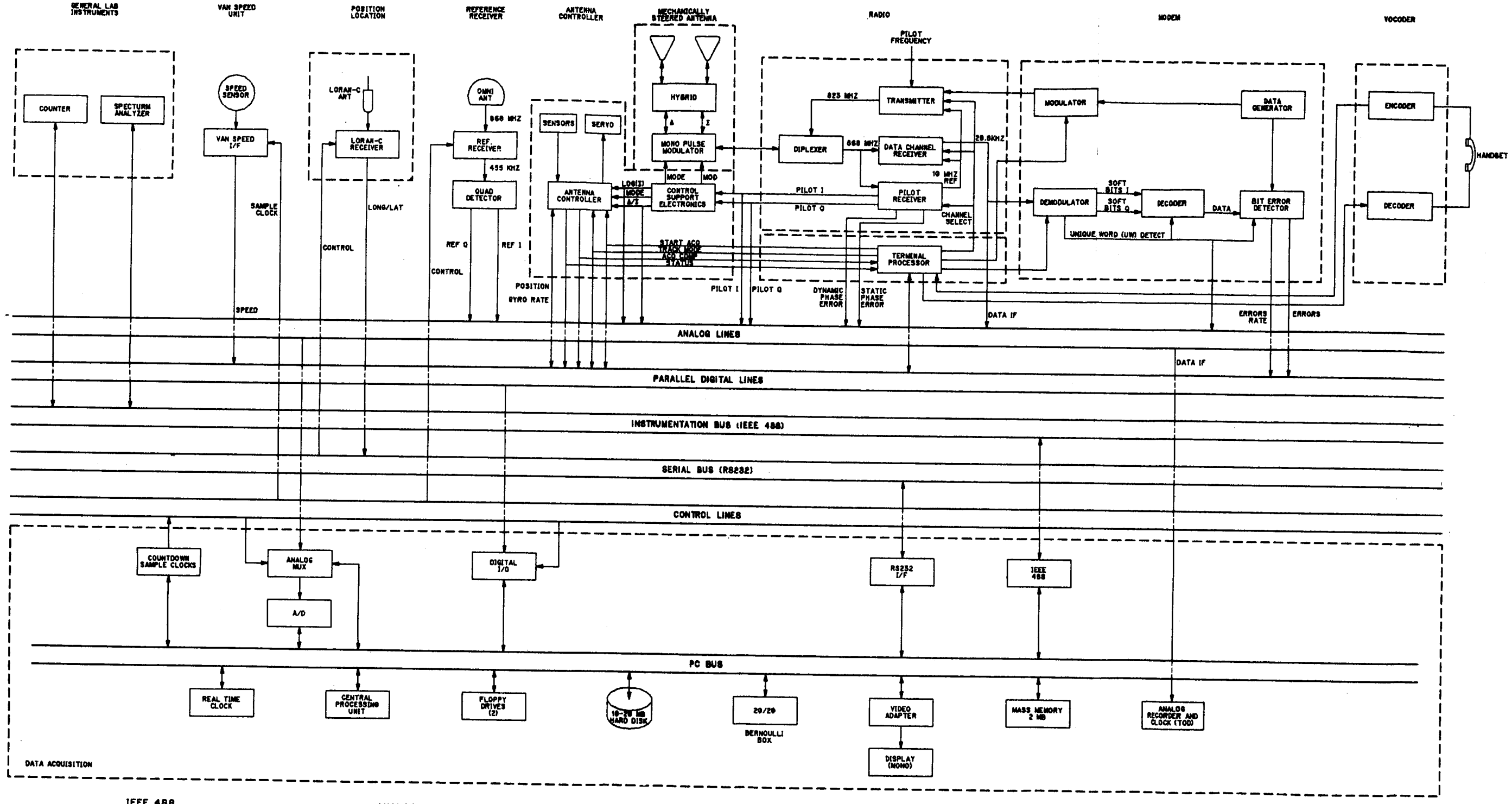
1) Antenna pointing experiment data: The purpose of this experiment was to quantify effects of electronic warmup, ambient temperature change, vehicle turn rate/acceleration, vehicle vibrations, and details of the tracking algorithm on the pointing performance of the antenna.

2) Radio experiment data: This experiment was useful in measuring the radio's performance in tracking the received pilot signal with or without Doppler. It was also used to measure the radio's performance in acquiring the signal with or without Doppler, and in conjunction with the antenna pointing system acquisition mode or tracking mode.

3) Modem experiment data: This experiment helped to determine the bit and packet error performance of the modem in the fading channel characteristic of the mobile satellite service. This experiment also measured the performance of the modem's unique word detection system.

4) Propagation channel characterization data: The data from this experiment is useful to characterize the communications channel in terms of fading and multipath losses.

The Data Acquisition System (DAS) consisted of an IBM AT-compatible microcomputer system, data acquisition boards, a keyboard, a color monitor, and finally a mass storage device. Figure 1 shows the DAS and all its interfaces with the experiments. The DAS operated on a continuous, real-time basis, gathering various data generated by the experiments described above. Each experiment produced different data to be recorded for post-analysis. Some of the data were analog in nature, while others were digital. The DAS was charged with the responsibility of converting the analog data into digital data by the usual methods of sampling and performing analog-to-digital (A/D) conversion. The output of the entire experiment was stored in a Bernoulli box (the mass storage device). Table 1 shows inputs from various data sources, analog as well as digital, along with the sample size required for the desired accuracy.



IEEE 488

ITEM	SPS RATE	SAMPLE SIZE
COUNTER	10	16 BITS

ANALOG

ITEM	SPS RATE	SAMPLE SIZE
DYNAMIC LOOP ERROR	33	12 BITS
STATIC LOOP ERROR	33	12
Δ/Σ	33	12
LOG(Σ)	33	12
REF I	1000	12
REF Q	1000	12
PILOT I	1000	12
PILOT Q	1000	12
TEMP	33	12

DIGITAL

ITEM	RATE SPS	SAMPLE SIZE
ANTENNA POSITION	66	16 BITS
GYRO RATE	66	16
EXPERIMENT MODE	33	16
VAN SPEED	33	16

RS232

ITEM	RATE SPS	SAMPLE SIZE
POSITION LAT	1	100 BITS
POSITION LONG	1	100

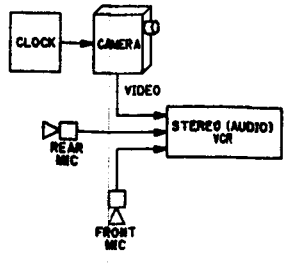


Figure 1. The DAS and its interfaces with the experiments

FOLDOUT FRAME

FOLDOUT FRAME

Table 1. Inputs from various data sources

ITEM	SAMPLING RATE (SAMPLES/SEC)	SAMPLE SIZE (BITS)	A=ANALOG DATA D=DIGITAL DATA
DYNAMIC LOOP ERROR	66.667	12	A
STATIC LOOP ERROR	66.667	12	A
Δ / Σ	100	12	A
LOG (Σ)	100	12	A
REFERENCE I CHANNEL	1000	12	A
REFERENCE Q CHANNEL	1000	12	A
PILOT I CHANNEL	1000	12	A
PILOT Q CHANNEL	1000	12	A
ANTENNA CONTROL	100	4	D
SPEED MEASUREMENT	1	16	D

The DAS sampled the various data and stored them in a binary offset format. This particular format is the result of the DAS software and is not common in computers. However, the DAS has the capability for converting the numbers from binary offset format to the more generally used ASCII format numbers. The data were divided into four different subsets according to their data rates and data-producing sources. The first set contains the reference and pilot channel in-phase and quadrature voltages. This set provides the experimenter with the amplitude and phase of the incoming reference and pilot channel signals. The sampling rates of the in-phase and quadrature channels were selected to be 1000 samples per second for the pilot as well as reference channels. This data set was named the "Propagation Data Set" for obvious reasons. The second data set consisted of data from the dynamic and static phase lock loop error of the receiver system, and this data set was termed the "Loop Data Set." The third set of data was termed the "Antenna Data Set" and it contained the normalized difference, log sum, and temperature. These data were from the mechanically steered antenna system designed by JPL. Each source in this data set was sampled at a rate of 100 samples per second. All the three sets of data described above contained analog data, i.e., the DAS sampled the output at the proper rates, performed an A/D operation on them, and finally stored the data in an appropriate data set. The fourth data set was termed the "Digital Data Set" because it stored data resulting from digital sources in the experiment. This set contained an azimuth position data file, a gyro data file, a van speed data file and, finally, a file termed "spare data." The spare data file was included to

make room for future expansion in digital sources. The digital data are sampled at relatively low rates.

The DAS was designed to accumulate 2.5 minutes of data from all four sets of data in its random access memory while simultaneously performing the A/D conversion for the analog sources. At the end of 2.5 minutes, the data were dumped into the mass storage device (the Bernoulli box). This action took some finite amount of time (much smaller than 2.5 minutes), during which the DAS did not gather any new data. After the data were dumped into the Bernoulli box, the 2.5 minutes of data accumulation started again. Using the data rates given in Table 1, one can compute that the 2.5 minutes of data accumulation from each set of data mean that there are 600,000 samples from the first set, 20,000 samples from the second set, 30,000 samples from the third set, and 50,000 samples from the fourth set. Thus, in all, there are 700,000 samples (numbers) in each 2.5 minutes' accumulation. This 2.5 minutes of data accumulation formed a file in the mass storage device. At the end of the experiment all the data were resident in numerous 2.5-minute files. Figure 2 shows the details of a typical 2.5-minute file from the mass storage device.

One thing to note here is that the DAS is a 16-bit machine, so when the sample needed only 12 bits to get the desired accuracy, the machine stored it as a 16-bit number with the four most significant bits turned to 0.

This section has described the data that were gathered as a result of the PiFEx experimentation; the following sections provide a possible data management system for the gathered data.

SECTION 3

TRANSFER OF DATA FROM THE MASS STORAGE DEVICE TO THE MINICOMPUTER

The DAS only sampled and accumulated the data into the mass storage device (Bernoulli box); it will not be used to analyze the data. This will be done by the VAX minicomputer. For this reason the data from the Bernoulli box will have to be transmitted to the VAX in some way. As was mentioned before, the data stored in the Bernoulli box is in a binary offset format; ordinarily a program would have to be written to convert this binary offset format to the ASCII format. However, to avoid this situation, the capability of the DAS to read and convert the numbers written in the binary offset format to the ASCII format will be used, and then the data will be sent to the VAX minicomputer via a hardwire connection. The DAS is equipped to do this easily.

As a result, we will assume in what follows that all the 2.5-minutes files have been duly converted to ASCII format and transported to the VAX by the process described above. We will also assume that the order in which the data were stored in the Bernoulli box did not change during the transportation from the microcomputer to the minicomputer. Consequently, Figure 2, which shows the 2.5-minute data file structure in the Bernoulli box, also shows the file structure in the VAX.

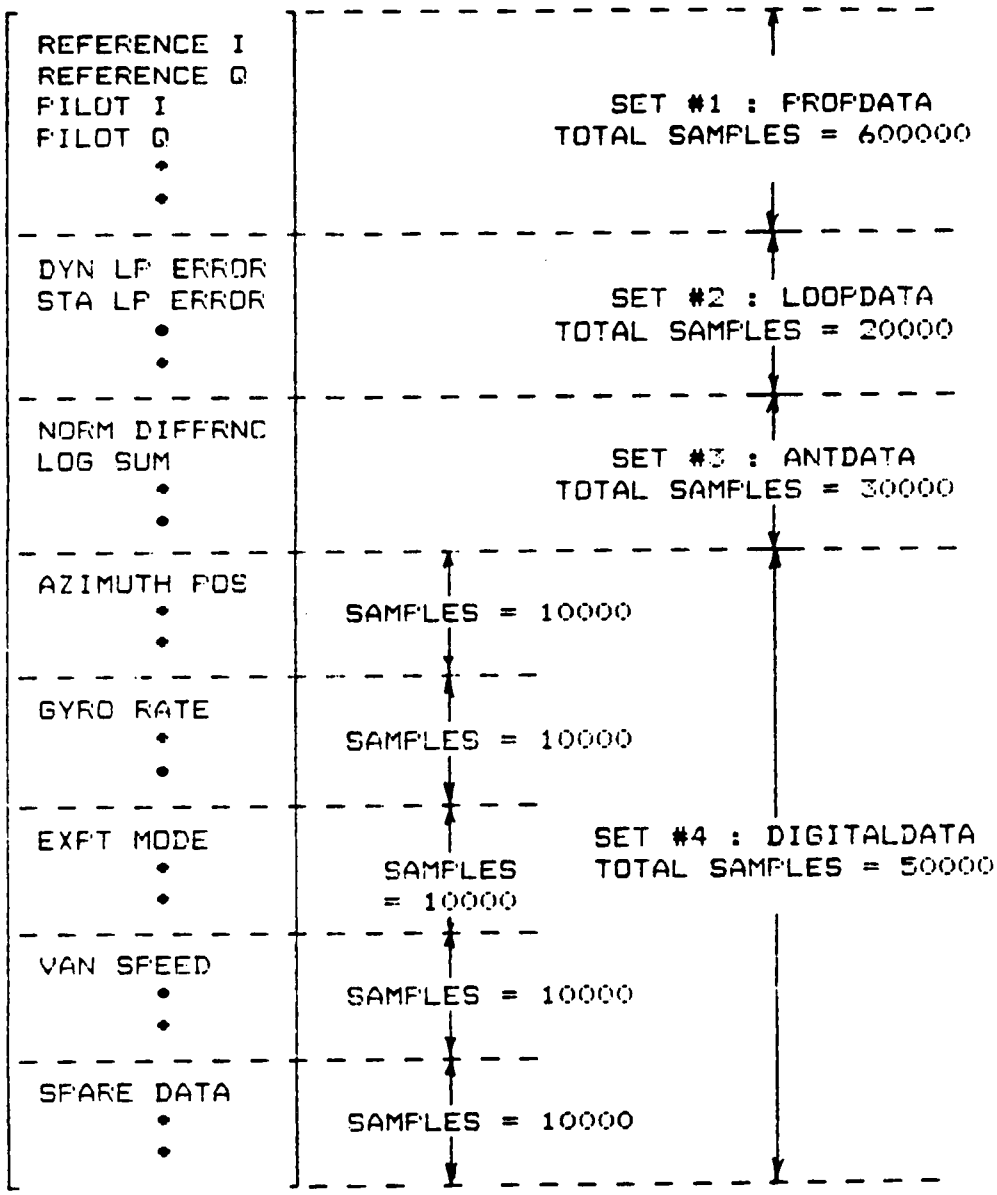


Figure 2. Format of a 2.5-minute file

SECTION 4

DATA BASE MANAGEMENT SYSTEM

The data base management system consists of several programs residing in the VAX minicomputer. These programs can be divided into two categories: one consisting of programs that handle the files and the other consisting of programs that handle the number-crunching aspects connected with the data. To use these files the user needs to log on to the VAX under his account code and then copy all the programs in the /U/KANTAK/PIFEX directory into his own directory. This is necessary to preserve the original files in case of a mishap with the user files. Once the data are transferred to the VAX minicomputer in the form of numerous 2.5-minute ASCII files, the above mentioned set of programs will be used to handle the data. These programs are listed below:

Programs for coarse handling of the 2.5-minute file data:

PIFEX1.F	Calibration program.
PIFEX2.F	Calibration program.
PIFEX3	File-handling program.
PIFEX4.F	Propagation data (set #1) handling program.
PIFEX5.F	Loop data (set #2) handling program.
PIFEX6.F	Antenna data (set #3) handling program.

Programs for propagation data analysis:

PIFEX10.F	Computes the probability of the signal

level's being greater than or equal to the abscissa.

PIFEX11.F Computes the probability density of signal levels and least square fits to the proper Rician density function.

PIFEX12.F Computes the frequency of the fade duration.

PIFEX13.F Creates a downloadable file for AKPLOT.

PIFEX14.F Computes the Fourier transform of 1024 samples.

The executable versions of the above files are included in the package so that the user does not have to compile anything. If the FORTRAN file is PIFEX##.F, then the corresponding executable file is called PIFEX##EX.

SECTION 5

CALIBRATION

The data that were sampled from the reference and pilot I and Q channels were essentially voltages, and these values depended upon the power supplied to the instruments. This implied that calibration of the instruments was necessary before data recording began. The following steps were followed to generate the calibration table to be used to scale the data:

1. Connect the directional coupler (or the switch located at the antenna terminal) of the reference receiver to the calibration signal source.

2. Vary the calibration signal power from -100 dBm to -70 dBm in steps of 2 dBm. The Data Acquisition System will sample the I and Q channels of the reference receiver once per step of the calibration signal, and the voltage levels will be stored in the Bernoulli box.

3. Connect the directional coupler (or the switch located at the antenna terminal) of the pilot receiver to the calibration signal source.

4. Follow the same step as for the reference receiver calibration.

5. Turn the switches on the reference and pilot receivers to their respective antennas.

At this point the van was ready to travel and take the readings for all the planned experiments. At the end of the experiments the calibration steps described were executed again to ensure that the parameters of the system had remained essentially constant. As a result of the calibration the following table (Table 2) was generated for both the reference and pilot channels. These two tables were stored in the Bernoulli box during the PiFEx test, and then they were transferred to the VAX minicomputer along with the rest of the data.

NOTE: The computer file storing this data will be a sequential file and each line of the above table will be stored as:

In-phase channel volts:

Quadrature channel volts:

Calibration signal power (dBm):

Table 2. Reference and pilot channel calibrations

CALIBRATION SIGNAL POWER (dBm)	IN PHASE (I) CHANNEL VOLTAGE (VOLTS)	QUADRATURE (Q) CHANNEL VOLTAGE (VOLTS)
-110		
-108		
-106		
-70		

5.1 CALIBRATION OF FILES

Calibration is needed only for the propagation data, i.e., for the first set of data in the 2.5-minute file (see Figure 2). There are 600,000 samples to be calibrated, and the remaining data in the 2.5-minute files (i.e., set #2, set #3 and set #4) should not be changed. The calibration tables for both the reference and pilot receivers were transmitted to the VAX computer and will be available to the user of the propagation data (the data of set #1).

5.2 PIFEX1.F

PURPOSE : This program allows the user to generate the calibration file if it was not generated at the experiment site, which may happen if there are any unforeseen experimental conditions at the site. If the calibration file was generated at the experiment site and was transferred to the VAX along with the data, then this program is not necessary.

USAGE : The symbolic file of the program is called PIFEX1.F. The extension "F" signifies that it is a FORTRAN file. The executable file of this program, called PIFEX1EX, is also provided so that the user does not have to compile it. To run this program the user should type

PIFEX1EX

and press return.

The following question and answer session will then begin:

PIFEX1 : Enter the desired calibration file name:

USER RESPONSE : The user should enter the desired name of the calibration file being generated. The user may have to run this program two times to generate the calibration files for the pilot and

reference channels.

NOTE : It should be noted by the user that if the user specifies an already existing file, then the system will rewrite on this file and the original data will be lost.

PIFEX1 : Enter the calibration test quantities one by one using "F" format. Inputting 0 for all entries will stop the program.

Enter sample number 1 quantities:

Power at the antenna terminal (dBm):

In-phase channel voltage (volts):

Quadrature channel voltage (volts):

USER RESPONSE : The user will input the power at the antenna terminal, the in-phase channel voltage, and the quadrature channel voltage, one by one in the real number format, i.e., ####.###. The decimal point has to be in every input explicitly. When the user has reached the end of his calibration table, he should enter a 0 for each entry, and this will stop the input process. No further user inputs are required.

NOTE : At this point PIFEX1 generates for each I and Q channel input a quantity vsqre, where

$$vsqre=(\text{I channel volts})^2 + (\text{Q channel volts})^2$$

and it stores I channel volts, Q channel volts, the power in dBm, and the vsqre in the desired calibration file named by the user earlier.

PIFEX1 OUTPUT : Showing the newly created file:

I channel volts	Q channel volts	Input power dBm	vi*vi+vq*vq
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.

5.3 PIFEX2.F

PURPOSE : This program calibrates the propagation data (data set #1) in any designated 2.5-minute file. This program does not change the data that follows the propagation data; i.e., data set #2, data set #3, and data set #4 remain unchanged because they do not have to be calibrated.

USAGE : The symbolic file of the program is called PIFEX2.F. The extension "F" signifies that it is a FORTRAN file. The executable file of this program, called PIFEX2EX, is also provided so that the user does not have to compile it. To run this program the user should type

PIFEX2EX

and press return.

The following question and answer session will then begin:

PIFEX2 : Enter the input file name:

USER RESPONSE : The user should enter the name of the 2.5-minute data file he desires to process (calibrate).

PIFEX2 : Enter the output file name:

USER RESPONSE : The user should enter the name of the desired

output file that will contain the calibrated data.

NOTE : It should be noted that if the name of the file specified is new, then a new file will be created, but if the file name is already in use, then the PIFEX2 program will overwrite the file, destroying its original contents.

PIFEX2 : Enter the pilot calibration file name:

USER RESPONSE : The user should enter the pilot calibration file name. This file either was generated using the PIFEX1 program or was generated on site and was transmitted to the VAX.

PIFEX2 : Enter the reference calibration file name:

USER RESPONSE : The user should enter the reference channel calibration file name. This file either was generated using the PIFEX1 program or was generated on site and was transmitted to the VAX.

PIFEX2 OUTPUT : The original 2.5-minute input file given to PIFEX2 program has in its data set #1 REF I, REF Q, PIL I, and PIL Q samples. There are 600,000 such samples. The output file will now contain

the REF SIG LEVEL (dBw), REF PHASE (rad), PIL SIG LEVEL (dBw), and PIL PHASE (rad). Please note that the signal levels are in dBw and not dBm. Data set #2, data set #3, and data set #4 in the original 2.5-minute file remain unchanged in the output file. Thus the size of the input file does not change; only the calibrated numbers are substituted for the raw numbers.

SECTION 6

FILE HANDLING

At this point the user has the 2.5-minute data files he is interested in calibrated and under the desired names. As was mentioned before, each calibrated 2.5-minute file has calibrated data set #1, data set# 2, data set #3, and data set #4. The user may be interested in any one or more of these data sets. The program PIFEX3 is supplied towards this end.

6.1 PIFEX3

PURPOSE : This program takes the calibrated or uncalibrated 2.5-minute file and segments it into eight different output files containing data set #1, data set #2, data set #3, and all the separate files of data set #4, respectively. The names of these eight files are : PROPDATA, LOOPDATA, ANTDATA, AZIDATA, GYRODATA, EXPTMODE, SPEEDATA, and SPARDATA.

USAGE : PIFEX3 is not a FORTRAN file; it is written in the UNIX system language. The program is an executable file, and to use it the user has to enter

PIFEX3 2.5-MINUTE FILE NAME

and press return.

PIFEX3 OUTPUT : The output of PIFEX3 will be the eight different files named above. The output file named "PROPDATA" will contain only the calibrated or uncalibrated propagation data (i.e., data set #1). "LOOPDATA" will contain only the dynamic loop error and the static loop error (i.e., data set #2). The "ANTDATA" will contain the normalized difference, the log sum, and temperature. Basically, "ANTDATA" has data set

#3. Finally, the data set #4 files are as follows: "AZIDATA", which contains the antenna azimuth position; "GYRODATA", which contains the vehicle turn rate; "EXPTMODE", which contains the information, valid or not; "SPEEDATA", which contains the van speed; and "SPARDATA", which will be the empty file provided for future expansion.

6.2 PIFEX4.F

PURPOSE : The file "PROPDATA" contains only the propagation data, with the four quantities of each sample of the reference and pilot receiver arranged in the following order: REF SIG LEVEL, REF SIG PHASE, PILOT SIG LEVEL, and PILOT SIG PHASE. It is anticipated that the researcher would like to have each of these four quantities in its own separate file.

USAGE : The "F" extension on the file name shows that the file is a FORTRAN file. The executable file is also given, under the name PIFEX4EX. To use this file, the user has to type

PIFEX4EX

and press enter.

The following question and answer session will then begin:

PIFEX4 : Enter the file name to store the ref signal levels:

USER RESPONSE : The user should enter a name for the file that will store the reference signal levels.

NOTE : If the name given is a new name (not already existing), then PIFEX4 will create the file, but

if the file already exists, then PIFEX4 will use this file and rewrite on the original data. This note will not be repeated for PIFEX4 and it should be assumed to be applicable whenever a file name is to be entered by the user.

PIFEX4 : Enter the file name to store the ref signal phases:

USER RESPONSE : Similar to the first response.

PIFEX4 : Enter the file name to store the pilot sig levels:

USER RESPONSE : Similar to the first response.

PIFEX4 : Enter the file name to store the pilot sig phases:

USER RESPONSE : Similar to the first response.

PIFEX4 OUTPUT : The data from the "PROPDATA" file will be divided into four files: the reference signal levels, reference signal phases, pilot signal levels, and pilot signal phases files. The names of these files, of course, will be user-supplied.

6.3 PIFEX5.F

PURPOSE : The file "LOOPDATA" contains only the loop data, which consists of measurements of the dynamic loop error and static loop error. It is anticipated that the researcher would like to have each of these two quantities in its own separate file.

USAGE : The "F" extension on the file name shows that the file is a FORTRAN file. The executable file is also given, under the name PIFEX5EX. To use this file, the user has to type

PIFEX5EX

and press enter.

The following question and answer session will then begin:

PIFEX5 : Enter the file name to store the dynamic loop error:

USER RESPONSE : The user should enter a name for the file that will store the dynamic loop error.

NOTE : If the name given is a new name (not already existing), then PIFEX5 will create the file, but if the file already exists, then PIFEX5 will use this file and rewrite on the original data. This

note will not be repeated for PIFEX5 and it should be assumed to be applicable whenever a file name is to be entered by the user.

PIFEX5 : Enter the file name to store the static loop error:

USER RESPONSE : Similar to the first response.

PIFEX5 OUTPUT : The data from the "LOOPDATA" file will be divided into two files: the dynamic loop error file and the static loop error file. The names of these files, of course, will be user-supplied.

6.4 PIFEX6.F

PURPOSE : The file "ANTDATA" contains the antenna data, which consists of measurements of the normalized difference, log sum, and temperature, arranged in that order. It is anticipated that the researcher would like to have each of these three quantities stated above in its own separate file.

USAGE : The "F" extension on the file name shows that the file is a FORTRAN file. The executable file is also given, under the name PIFEX6EX. To use this file, the user has to type

PIFEX6EX

and press enter.

The following question and answer session will then begin:

PIFEX6 : Enter the file name to store the normalized difference:

USER RESPONSE : The user should enter a name for the file that will store the normalized difference.

NOTE : If the name given is a new name (not already existing), then PIFEX6 will create the file, but if it already exists, then PIFEX6 will use this

file and rewrite on the original data. This note will not be repeated for PIFEX6 and it should be assumed to be applicable whenever a file name is to be entered by the user.

PIFEX6 : Enter the file name to store the log sum data:

USER RESPONSE : Similar to the first response.

PIFEX6 : Enter the file name to store the temperature:

USER RESPONSE : Similar to the first response.

PIFEX6 OUTPUT : The data from "ANTDATA" file will be divided into three files: the normalized difference, log sum, and temperature. The names of these files, of course, will be user-supplied.

SECTION 7

PROGRAMS FOR PROPAGATION DATA ANALYSIS

Every 2.5-minute file has the propagation data in data set #1. The programs described in Section 6 will separate the propagation data set into its own file. Once the propagation data are isolated, the researcher may like to perform the usual operations on these data, such as computing the probability of the signal level's being greater than or equal to the given level, computing the probability density of the signal levels, computing the frequency of the fade duration, and generating the Fourier transform of the signal levels. Towards that end, a few programs are provided as noted in Section 4.

Most operations on data described in this section generate the data for presentation in a plot form and hence make it necessary to have a plotter routine either on the VAX or on the researcher's microcomputer. A new plotter routine called AKPLOT goes a long way towards meeting these plotting needs. AKPLOT is both completely interactive and user-friendly. At the beginning of every plot there is an interactive session through which AKPLOT gathers the parameters necessary for the plot. The plot is produced on the screen, and the user is given the ability to change the plot to the desired form and finally perform a screen dump to a printer to produce a hard copy. AKPLOT was used to produce all the graphs in this manual.

7.1 PIFEX10.F

PURPOSE : This program is useful in producing data for computing the probability of the signal level's being greater than or equal to the specified signal level.

NOTE : The program specifies the signal level, starting at -130 dBw and going to -80 dBw in steps of 0.5 dBw. For every step the probability of the signal level in the given file exceeding the specified level is computed and stored.

USAGE : The symbolic file of the program is called PIFEX10.F. The extension "F" signifies that it is a FORTRAN file. The executable file for this program is also provided and is named PIFEX10EX. To run this program the user should type

PIFEX10EX

and press return.

The following question and answer session will then follow:

PIFEX10 : Enter the input signal levels file name:

USER RESPONSE : The user is required to input the name of the signal levels file for either the reference receiver or the pilot receiver. These files were

made by use of the PIFEX4 program.

PIFEX10 : Enter the output file name:

USER RESPONSE : The user is required to enter the name of the output file.

PIFEX10 OUTPUT : The input to this program is the signal levels file, while the output will be two vectors. The first vector will contain the signal levels, while the second one will contain the corresponding probability of the incoming signal level's being greater than or equal to the level stored in the first vector. The levels in vector one will start at -130.0 dBw and reach up to -80.0 dBw in steps of 0.5 dBw. Thus there will be 100 entries in each vector. Figure 3 shows a representative output of this program.

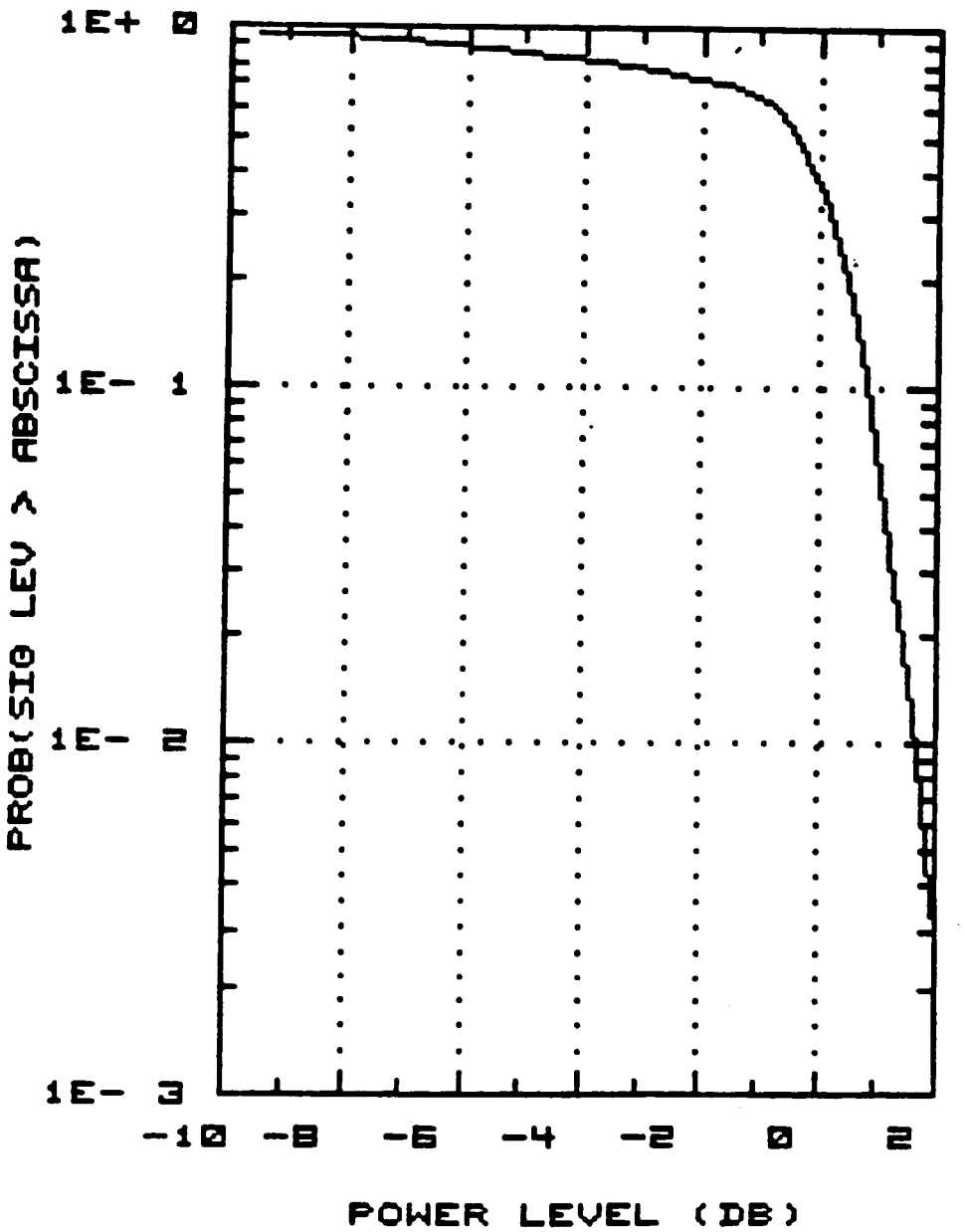


Figure 3. Representative output of the PIFEX10 program

7.2 PIFEX11.F

PURPOSE : This program computes the probability density of the signal levels and fits a Rician density function of the proper K factor to the data points via least squares fitting.

USAGE : The symbolic file is PIFEX11.F. As before, the extension "F" indicates that the file is a FORTRAN file. The compiled (executable) version is supplied in the file PIFEX11EX. To use this program the user is required to input

PIFEX11EX

and press return.

The following question and answer session will then begin:

PIFEX11 : Enter the input signal levels file name:

USER RESPONSE : The user is required to input the name of the file storing the signal levels. This file should be generated by using the PIFEX4 program.

PIFEX11 : Enter the output file name:

USER RESPONSE : The user is required to input the name of the file in which the output will be stored. If the entered name is a new name, then this file will

be created, but if the name is already in use, then PIFEX11 will use this file, i.e., the original data of the file will be erased.

PIFEX11 OUTPUT : The input to this program is the signal level file. The output file, under the user-supplied name, will contain 100 entries of x(i),y(i) pairs where x(i) will contain the signal level starting from -130.0 dBw and reaching -80.0 dBw in steps of 0.5 dBw, while y(i) will contain the corresponding probability density value. The next 100 entries in this file will contain the pair x(i),y(i) where the x(i) value is the same as described above, but the y(i) value will be the corresponding Rician density value. Thus there will be in all 200 rows of entries into this file. This information will be needed by the plotter routine to be used to plot the curves. Figure 4 shows a representative output.

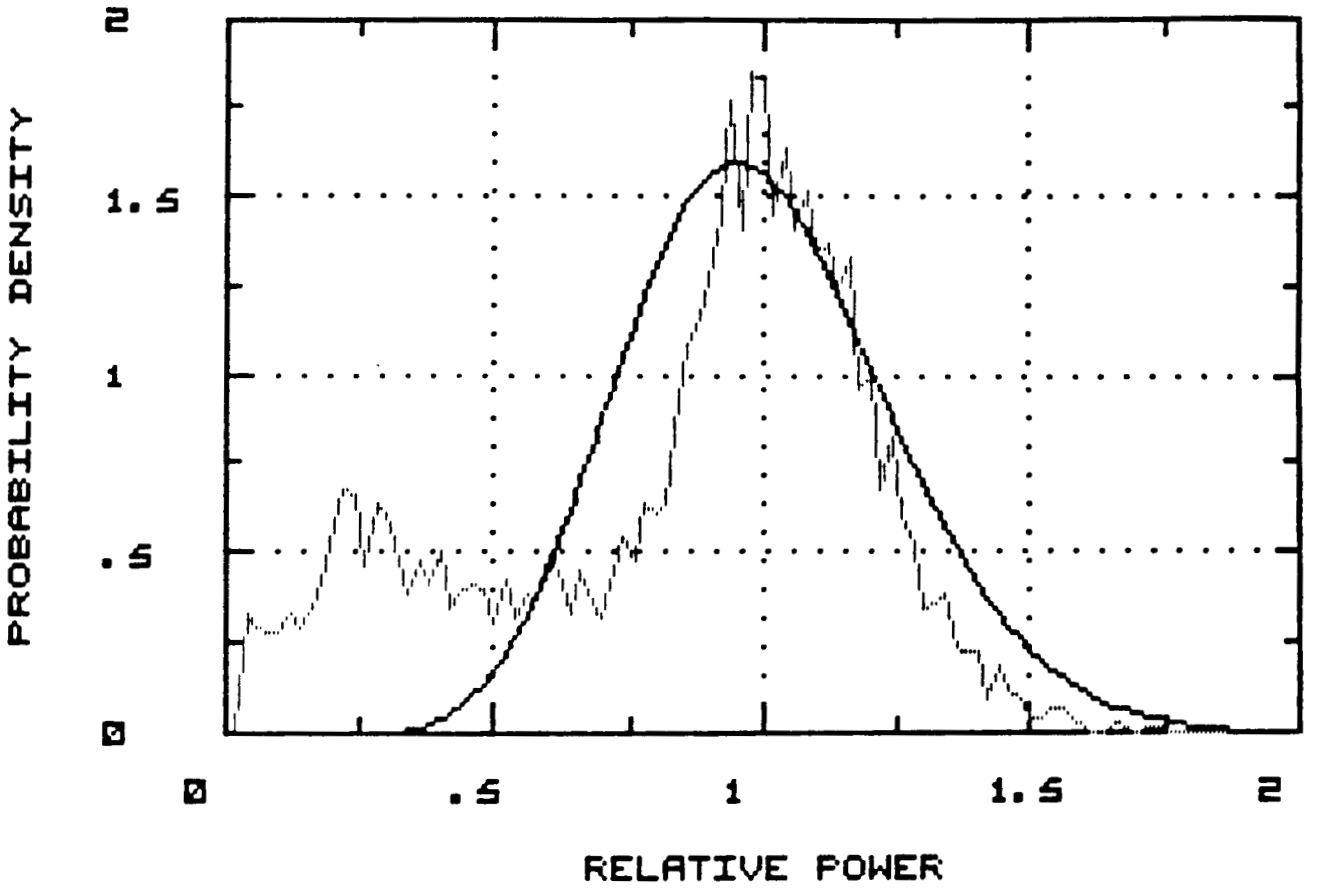


Figure 4. Representative output of the PIFEX11 program

7.3 PIFEX12.F

PURPOSE : This program computes the frequency of the input data's fade duration.

USAGE : The symbolic file is PIFEX12.F. As before, the extension "F" indicates that the file is a FORTRAN file. The compiled (executable) version is supplied in the file PIFEX12EX. To use this program the user is required to input

PIFEX12EX

and press return.

The following question and answer session will then begin:

PIFEX12 : Enter the input signal levels file name:

USER RESPONSE : The user is required to input the name of the file storing the signal levels. This file should be generated by using the PIFEX4 program.

PIFEX12 : Enter the output file name:

USER RESPONSE : The user is required to input the name of the file in which the output will be stored. If the entered name is a new name, then this file will be created, but if the name is already in use,

then PIFEX12 will use this file, i.e., the original data of the file will be erased.

PIFEX12 : Enter the desired signal level:

USER RESPONSE : This is the signal level used for the fade duration.

PIFEX12 OUTPUT : The sampling interval is predefined to be 1/1000 of a second, i.e., between every two samples there is 0.001 seconds of difference. The input to this program is the signal levels file. The output will be a file containing two vectors, x(i) and y(i); i=1,100. The first entry, x(i), will be the bin number giving the fade duration in terms of the samples (e.g., bin number 5 will refer to the fade that lasts 5 samples' time, i.e., 5*0.001 seconds) and y(i) will give the number of times this fade takes place in the input file. Figure 5 shows a representative output of this program.

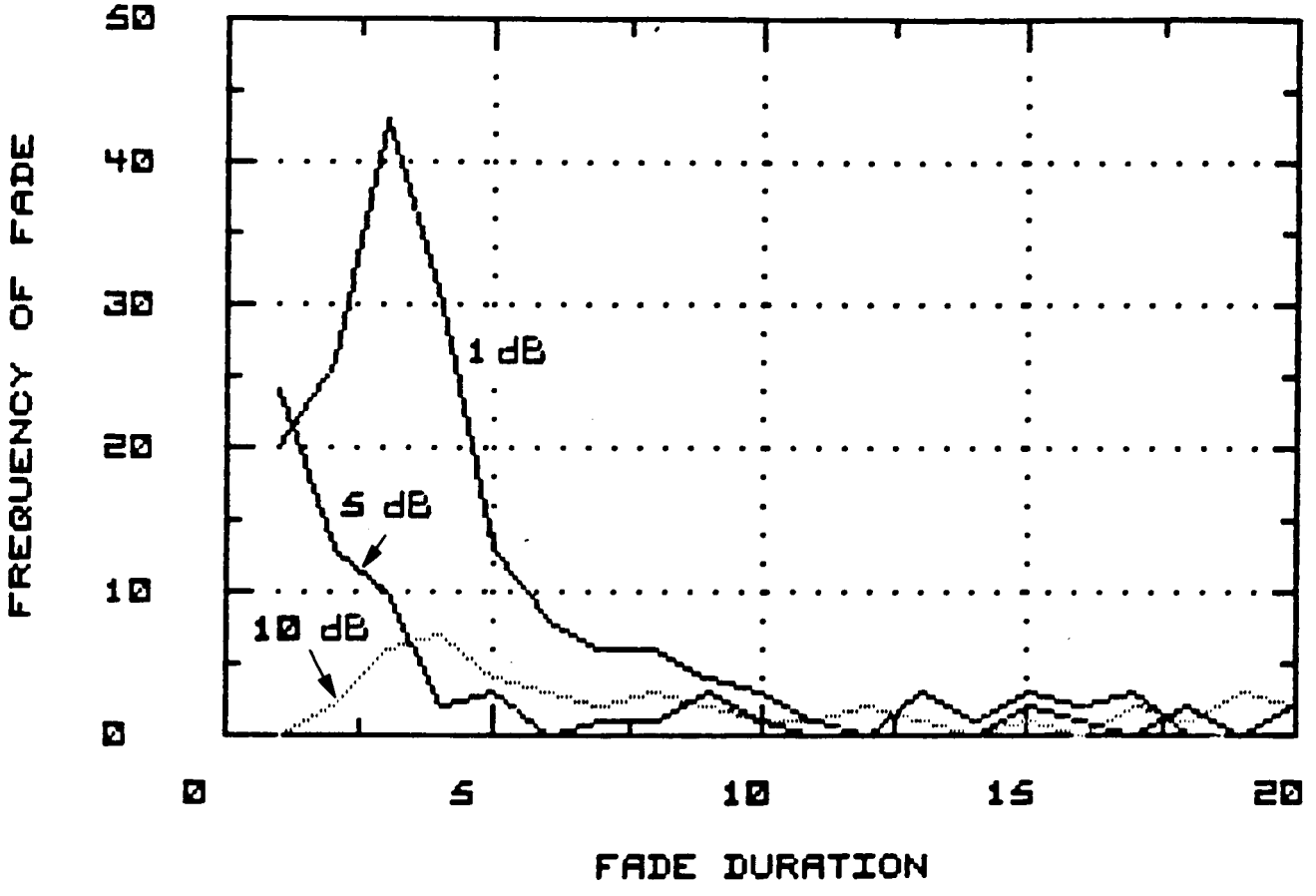


Figure 5. Representative output of the PIFEX12 program

7.4 PIFEX13.F

PURPOSE : This program generates a downloadable file out of the signal levels file and the signal phases file for the use of the plotter routine AKPLOT.

USAGE : The symbolic file is supplied in the file PIFEX13.F. The extension "F" signifies that the file is a FORTRAN file. The compiled version (executable version) is also supplied and is under the name PIFEX13EX. To use this program the user has to type

PIFEX13EX

and press return.

The following question and answer session will then begin:

PIFEX13 : Enter the input file name:

USER RESPONSE : The user is expected to enter the name of the file that contains the data that needs to be plotted using the plotter routine AKPLOT.

PIFEX13 : Enter the output file name:

USER RESPONSE : The user should enter the name of the file under which the data prepared for AKPLOT will be stored. If the name supplied to PIFEX13 is a new

name (i.e., no file exists under this name), then PIFEX13 will create the file and then store the output data in that file. If the supplied name exists in the directory, then PIFEX13 will erase the data in there and replace it with new data.

PIFEX13 OUTPUT : The input to this program is the file containing the signal levels, and the output of this program will be a file that will contain the pair x(i),y(i) where x(i) will store the time instant, i.e., the sample number*0.001, and the y(i) will contain the signal level. There will be only 1000 entries in this file. Figures 6 and 7 show plots of the signal amplitude and phase representative of plots using the AKPLOT routine.

NOTE : This program may be altered to produce other plots using AKPLOT. One may access the symbolic file and change the x(i) computation to whatever is to be plotted on the x axis, and y(i) may be read from the supplied file. Also, one may change the beginning of the output file to any time instant by jumping a suitable number of samples in the input file.

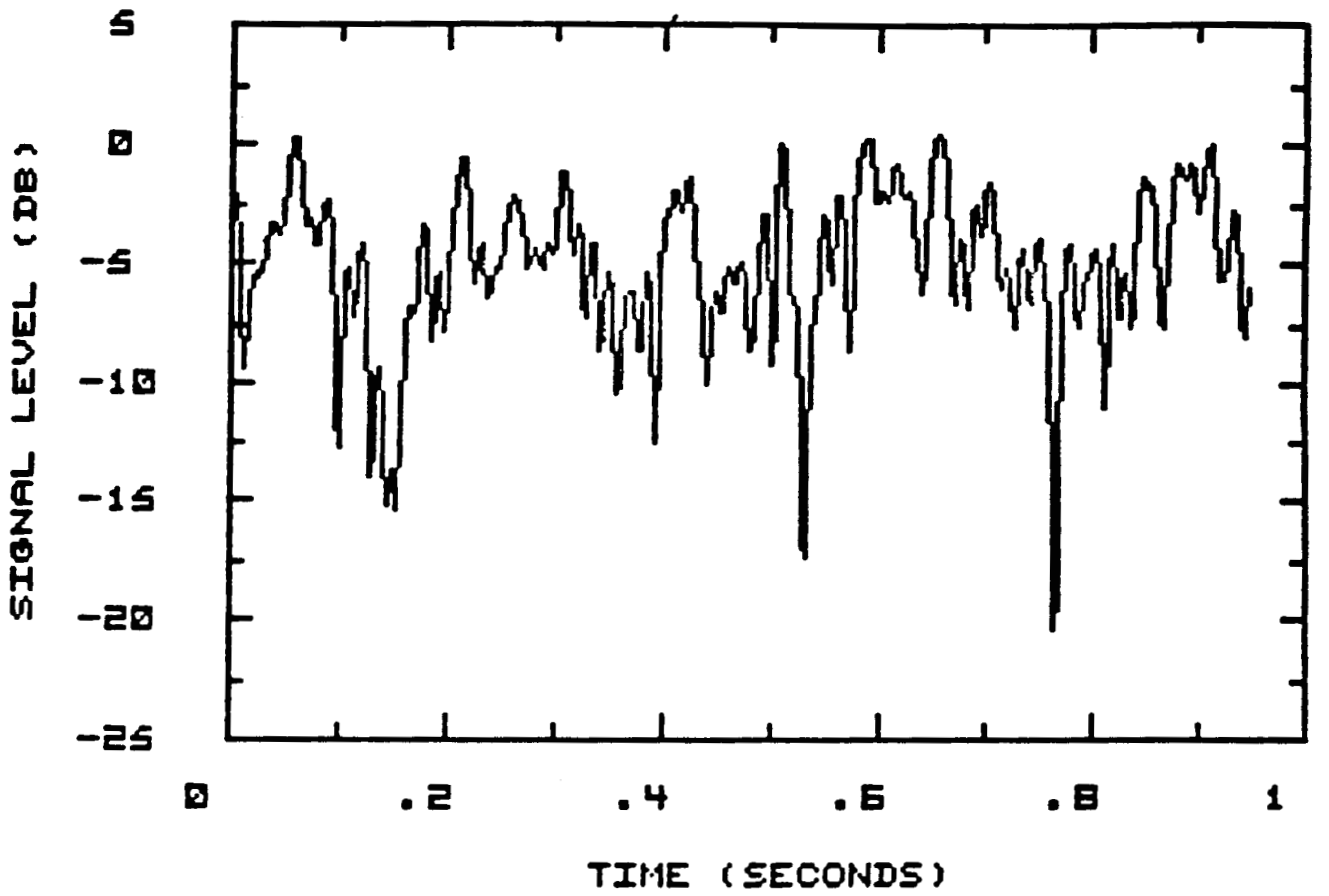


Figure 6. Representative plot of the signal amplitude using the AKPLOT routine

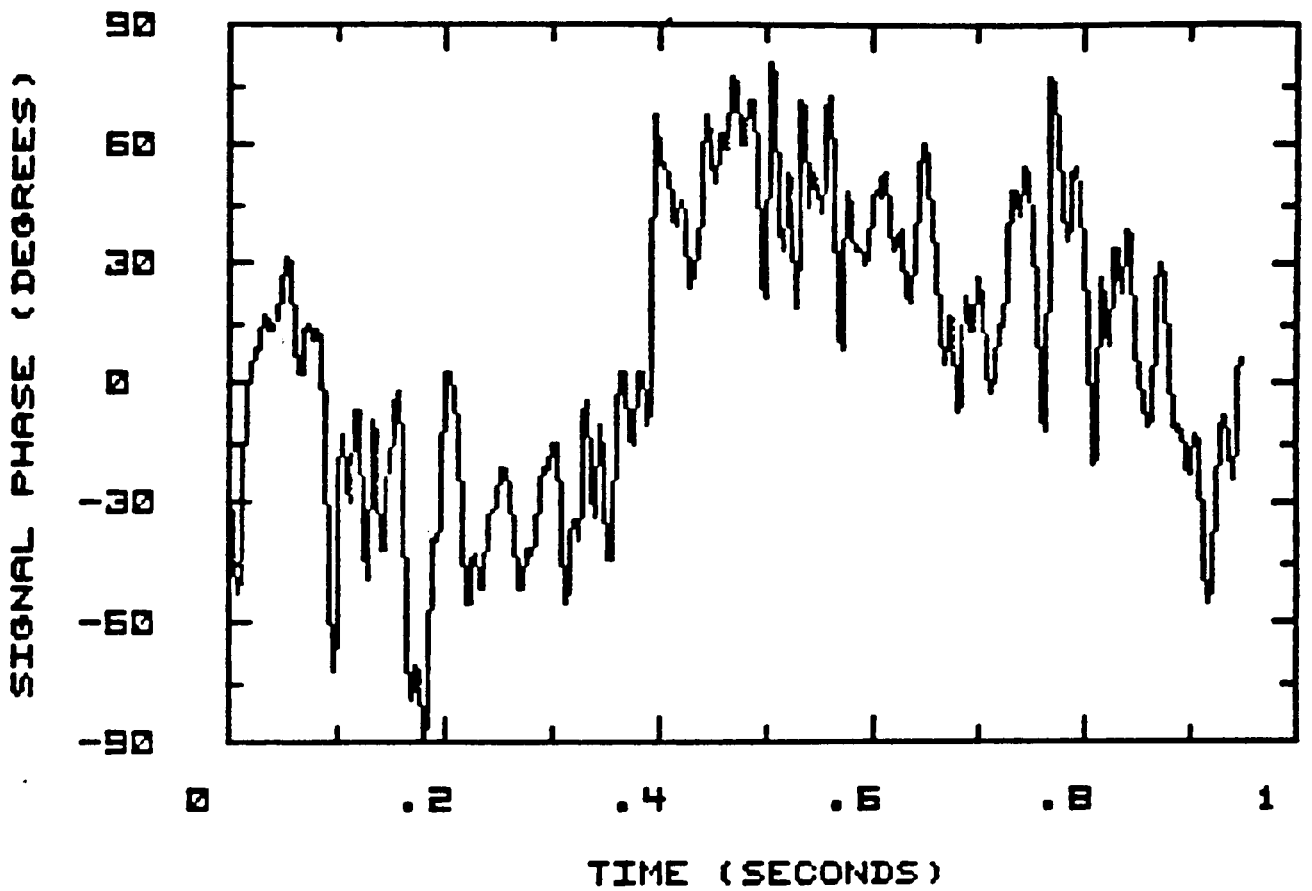


Figure 7. Representative plot of the signal phase using the AKPLOT routine

7.5 PIFEX14.F

PURPOSE : This program computes the Fourier transform of the first 1024 samples of the input file.

USAGE : The symbolic file is supplied in the file PIFEX14.F. The extension "F" signifies that the file is a FORTRAN file. The compiled version (executable version) is also supplied, under the name PIFEX14EX. To use this program the user has to type

PIFEX14EX

and press return.

The following question and answer session will then begin:

PIFEX14 : Enter the input file name:

USER RESPONSE : The user is expected to enter the name of the file that contains the data that needs to be Fourier transformed.

PIFEX14 : Enter the output file name:

USER RESPONSE : The user should enter the name of the file under which the Fourier-transformed data will be stored. If the name supplied to PIFEX14 is a new name (i.e., no file exists under this name),

then PIFEX14 will create the file and then store the output data in that file. If the supplied name exists in the directory, then PIFEX14 will erase the data in there and replace it with new data.

PIFEX14 OUTPUT : The input to this program is the file containing the signal levels, and the output of this program will be a file that will contain the pair x(i),y(i) where x(i) will store the frequency and y(i) will contain the corresponding magnitude of the signal level's Fourier transform. There will be only 1000 entries out of the possible 1024 entries in this file. Figure 8 plots the frequency on the x axis and the magnitude of the Fourier transform on the y axis using the AKPLOT routine.

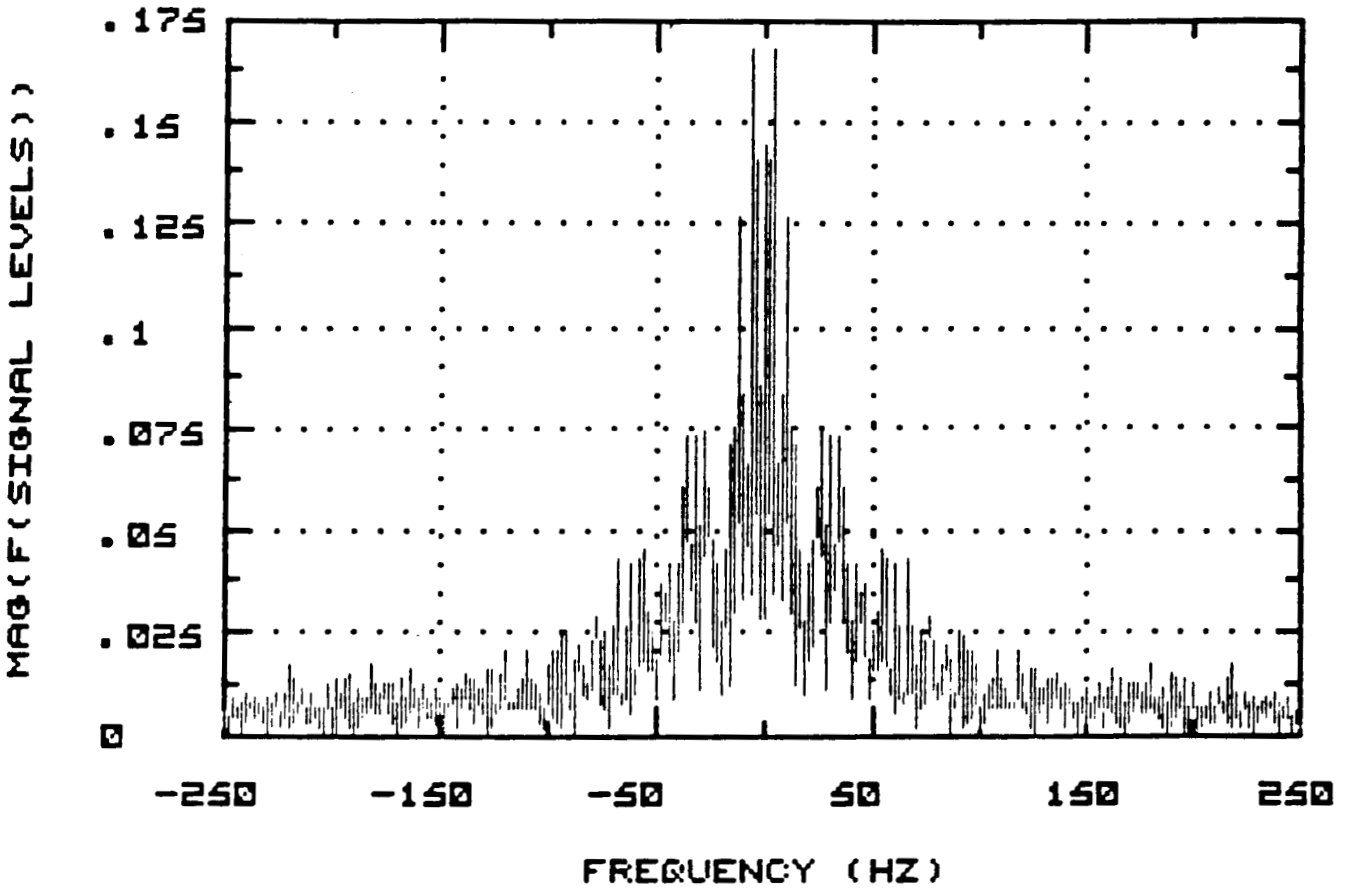


Figure 8. Representative plot of the frequency and magnitude of the Fourier transform using the AKPLOT routine

APPENDIX

Symbolic File PIFEX1.F

```
c This program creates the necessary calibration file
c by manually inputting the data noted down at the test
c site during calibration.
c
c
          character fname*12
          logical used
          write(6,200)
200       format("Enter the desired calibration file name :
          ")
300        format(f20.10)
          read(5,201) fname
200       Format(a12)
          inquire(file=fname,exist=used)
          if(.not.used)goto 211
          open(unit=11,file=fname,status="old")
          goto 212
211        open(unit=11,file=fname,status="new")
212       write(6,202)
202       format("  ")
          i=1
          write(6,101)
101       format("Enter the calibration test quantities one
          by one")
          write(6,102)
102       format("using f format all entries 0 will stop
          the program :")
100       write(6,202)
          write(6,202)
          write(6,203) i
203       format("Enter sample number ",i3," quantities :")
          write(6,202)
          write(6,204)
204       format("Power at the antenna terminal (dBm) : ")
          read(5,300) dbm
          write(6,205)
205       format("In phase channel voltage (volts) : ")
          read(5,300)vi
          write(6,206)
206       format("Quadrature channel voltage (volts) : ")
          read(5,300) vq
          if(vi.eq.0.0.and.vq.eq.0.0.and.dBm.eq.0.0)goto 50
          write(11,300) vi
          write(11,300) vq
          write(11,300) dbm
          vsqre=vi*vi+vq*vq
          write(11,300) vsqre
          i=i+1
          goto 100
 50       close(11)
          open(unit=11,file=fname,status="old")
```

```
            write(6,202)
            write(6,209) fname
209         format("Showing the newly created file : ",a12)
            write(6,202)
            write(6,202)
            write(6,207)
207         format(7x,"I channel volts",5x,"Q channel volts",
      c     5x,"input power (dBm)",3x,"vi*vi + vq*vq")
            write(6,202)
400         read(11,300,end=60) a
            read(11,300) b
            read(11,300) c
            read(11,300) d
            write(6,208) a,b,c,d
            goto 400
208         format(3x,f17.8,3x,f17.8,3x,f17.8,3x,f17.8)
 60         stop
            end
```

Symbolic File PIFEX2.F

```
c This program calibrates the test data points for the
c  propagation test (first 600,000 entries in the incoming
c  file) and produces a new file with the calibrated dBw and
c  phases of the signal points and the remaining data points
c  remain unchanged.
c
            dimension xp(1000),yp(1000),xr(1000),yr(1000)
            logical used
            character finp*12,fout*12,calp*12,calr*12
            write(6,200)
200         format("Enter the input file name : ")
            read(5,201)finp
201         format(a12)
            write(6,202)
202         format("  ")
            write(6,203)
203         format("Enter the output file name : ")
            read(5,201)fout
            write(6,202)
            open(unit=11,file=finp,status="old")
            inquire(file=fout,exist=used)
            if(.not.used)goto 104
            open(unit=12,file=fout,status="old")
            goto 105
104         open(unit=12,file=fout,status="new")
105         i=1
            j=0
            write(6,204)
204         format("Enter the pilot calibration file name : ")
            read(5,201)calp
            open(unit=13,file=calp,status="old")
            write(6,202)
            write(6,205)
205         format("Enter the reference calibration file name : ")
          read(5,201)calr
          open(unit=14,file=calr,status="old")
300       format(f20.10)
301       read(13,300,end=302)a
          read(13,300)b
          read(13,300)yp(i)
          yp(i)=yp(i)-30.0
          xp(i) = a*a+b*b
          read(14,300)a
          read(14,300)b
          read(14,300)yr(i)
          yr(i)=yr(i)-30.0
          xr(i) = a*a+b*b
          i=i+1
          goto 301
302       n=i-1
          i=1
```

```
500         read(11,300)xri
            read(11,300)xrq
            read(11,300)xpi
            read(11,300)xpq
            siglevr=xri*xri+xrq*xrq
            siglevp=xpi*xpi+xpq*xpq
            sigphsr=atan2(xrq,xri)
            sigphsp=atan2(xpq,xpi)
            if(siglevr.lt.xr(1))then
            siglevr=xr(1)
            goto 210
            endif
            if(siglevr.gt.xr(n))then
            siglevr=xr(n)
            goto 210
            endif
            if(siglevp.lt.xp(1))then
            siglevp=xp(1)
            goto 210
            endif
            if(siglevp.gt.xp(n))then
            siglevp=xp(n)
            goto 210
            endif
            do 206 k=1,n-1
            if(siglevr.ge.xr(k).and.siglevr.le.xr(k+1))goto 207
206         continue
207         siglevr=yr(k)+((yr(k+1)-yr(k))/(xr(k+1)-xr(k)))*
        c   (siglevr-xr(k))
            do 208 k=1,n-1
            if(siglevp.ge.xp(k).and.siglevp.le.xp(k+1))goto 209
208         continue
209         siglevp=yp(k)+((yp(k+1)-yp(k))/(xp(k+1)-xp(k)))*
        c   (siglevp-xp(k))
210         continue
            if(i.eq.1000)j=j+1
            if(j.eq.150)goto 211
            if(i.eq.1000)i=0
            i=i+1
            write(12,300)siglevr
            write(12,300)sigphsr
            write(12,300)siglevp
            write(12,300)sigphsp
            goto 500
211         read(11,300,end=212)a
            write(12,300)a
            goto 211
212         stop
            end
```

Symbolic File PIFEX3

```
echo
echo OLD FILE REMOVAL PROCEDURE INITIATED
echo
number=0
if test -s propdata
then rm propdata
number=1
echo
echo removing old propdata file
fi
if test -s loopdata
then rm loopdata
number=1
echo
echo removing old loopdata file
fi
if test -s antdata
then rm antdata
number=1
echo
echo removing old antdata file
fi
if test -s azidata
then rm azidata
number=1
echo
echo removing old azidata file
fi
if test -s gyrodata
then rm gyrodata
number=1
echo
echo removing old gyrodata file
fi
if test -s exptmode
then rm exptmode
number=1
echo
echo removing old exptmode file
fi
if test -s speedata
then rm speedata
number=1
echo
echo removing old speedata file
fi
if test -s spardata
then rm spardata
number=1
echo
echo removing old spardata file
```

```
fi
if test $number -eq 0
then echo
echo no old files to be removed
fi
echo
echo
echo NEW FILE CREATION PROCEDURE INITIATED
split -600000 $1
echo
mv xaa propdata
num='wc -l < propdata'
echo
echo propdata file was created and it contain
$num lines of data
mv xab residue
split -50000 residue
rm residue
mv xaa residue1
mv xab residue2
split -20000 residue1
rm residue1
mv xaa loopdata
num='wc -l < loopdata'
echo
echo loopdata file was created and it contains
$num lines of data
cat xac>>xab
rm xac
mv xab antdata
num='wc -l < antdata'
echo
echo antdata file was created and it contains
$num lines of data
split -10000 residue2
rm residue2
mv xaa azidata
num='wc -l < azidata'
echo
echo azidata file was created and it contains
$num lines of data
mv xab gyrodata
num='wc -l < gyrodata'
echo
echo gyrodata file was created and it contains
$num lines of data
mv xac exptmode
num='wc -l < exptmode'
echo
echo exptmode file was created and it contains
$num lines of data
mv xad speedata
num='wc -l <speedata'
echo
echo speedata file was created and it contains
```

```
$num lines of data
mv xae spardata
num='wc -l < spardata'
echo
echo spardata file was created and it contains
$num lines of data
echo
echo
echo TASK COMPLETED
echo
echo
echo
```

Symbolic File PIFEX4.F

```
c This program gets the reference channel signal levels
c and phase separated from the pilot channel signal levels
c and phase in the file "propdata". Propdata is the name
c of one of the output files of the program "pifex3".
c
c
          logical used
          character fa*12,fb*12,fc*12,fd*12
          open(unit=15,file="propdata",status="old")
          write(6,200)
200       format("Enter the file name to store ref signal
          levels :")
          read(5,201)fa
201       format(a12)
          inquire(file=fa,exist=used)
          if(.not.used)goto 104
          open(unit=11,file=fa,status="old")
          goto 105
104       open(unit=11,file=fa,status="new")
105       write(6,202)
202       format("  ")
          write(6,203)
 203      format("Enter the file name to store ref signal
          phases :")
          read(5,201)fb
          inquire(file=fb,exist=used)
          if(.not.used)goto 106
          open(unit=12,file=fb,status="old")
          goto 107
106       open(unit=12,file=fb,status="new")
107       write(6,202)
          write(6,204)
204       format("Enter the file name to store pilot sig
          levels :")
          read(5,201)fc
          inquire(file=fc,exist=used)
          if(.not.used)goto 108
          open(unit=13,file=fc,status="old")
          goto 109
108       open(unit=13,file=fc,status="new")
109       write(6,202)
          write(6,205)
205       format("Enter the file name to store pilot sig phases :")
        read(5,201)fd
        inquire(file=fd,exist=used)
        if(.not.used)goto 110
        open(unit=14,file=fd,status="old")
        goto 111
110     open(unit=14,file=fd,status="new")
111     write(6,202)
300     format(f20.10)
```

```
      do 600 i=1,60
600   read(15,300)a
400   read(15,300,end=500)a
      write(11,300)a
      read(15,300)b
      write(12,300)b
      read(15,300)c
      write(13,300)c
      read(15,300)d
      write(14,300)d
      goto 400
500   continue
      stop
      end
```

Symbolic File PIFEX5.F

```
c  This program separates the "loopdata" file into two
c  separate files containing dynamic loop error and static
c  loop error "loopdata" is the name of one of the output
c  files of the program PIFEX3.
c
           logical used
           character fa*12,fb*12
           open(unit=15,file="loopdata",status="old")
           write(6,200)
200        format("Enter the file name to store dynamic
           loop error :")
           read(5,201)fa
201        format(a12)
           inquire(file=fa,exist=used)
           if(.not.used)goto 104
           open(unit=11,file=fa,status="old")
           goto 105
104        open(unit=11,file=fa,status="new")
105        write(6,202)
202        format("  ")
           write(6,203)
203        format("Enter the file name to store static
           loop error :")
           read(5,201)fb
           inquire(file=fb,exist=used)
           if(.not.used)goto 106
           open(unit=12,file=fb,status="old")
           goto 107
106        open(unit=12,file=fb,status="new")
107        write(6,202)
300        format(f20.10)
400        read(15,300,end=500)a
           write(11,300)a
           read(15,300)b
           write(12,300)b
           goto 400
500        continue
           stop
           end
```

Symbolic File PIFEX6.F

```
c This program separates the "antdata" file into three
c separate files containing normalized difference , log sum
c and finally the temperature. "antdata" is the name of one
c of the output files of the program PIFEX3.
c
          logical used
          character fc*12,fd*12,fe*12
201       format(a12)
          write(6,202)
202       format("  ")
          open(unit=15,file="loopdata",status="old")
          write(6,204)
204       format("Enter the file name to store normalized
          difference :")
          read(5,201)fc
          inquire(file=fc,exist=used)
          if(.not.used)goto 108
          open(unit=13,file=fc,status="old")
          goto 109
108       open(unit=13,file=fc,status="new")
109       write(6,202)
          write(6,205)
205       format("Enter the file name to store log sum :")
          read(5,201)fd
          inquire(file=fd,exist=used)
          if(.not.used)goto 110
          open(unit=14,file=fd,status="old")
          goto 111
110       open(unit=14,file=fd,status="new")
111       write(6,202)
          write(6,206)
206       format("Enter the file name to store
          temperature :")
          read(5,201)fe
          inquire(file=fe,exist=used)
          if(.not.used)goto 113
          open (unit=12,file=fe,status="old")
          goto 114
113       open(unit=12,file=fe,status="new")
114       write(6,202)
300       format(f20.10)
400       read(15,300,end=500)c
          write(13,300)c
          read(15,300)d
          write(14,300)d
          read(15,300)e
          write(12,300)e
          goto 400
500       continue
          stop
          end
```

Symbolic File PIFEX10.F

```
c this program computes the probability of signal level
c being greater of equal to the abscissa.
           dimension x(100),y(100)
           logical used
           character fname*12,fnam*12
           write(6,200)
200        format(' enter the input signal levels
           file name ')
           read(5,100)fname
           write(6,301)
301        format(' enter the output file name')
           read(5,100)fnam
100        format(a12)
           open(unit=11,file=fname,status='old')
           inquire(file=fnam,exist=used)
           if(.not.used)goto 700
           open(unit=12,file=fnam,status="old")
           goto 701
700        open(unit=12,file=fnam,status='new')
701        do 5 i=1,100
           ii=0
           j=0
           x(i)=-130.0+(i-1)*0.5
 3         read(11,777,end=4) buf
           if(buf.ge.x(i)) j=j+1
           ii=ii+1
           goto 3
 4         rewind(11)
           y(i)=float(j)/float(ii)
           write(12,7)x(i),y(i)
           write(6,7)x(i),y(i)
 5         continue
 6         format(i16)
777        format(f20.10)
 7         format(f20.10,1x,f20.10)
           stop
           end
```

Symbolic File PIFEX11.F

```
c this programm computes the probability density of the
c signal levels and least squares fits it to a proper rician
c density function.
            double precision z(100),x(100),y(100)
            double precision xx,step,pp,sum,sum1,del
            double precision fun
            logical used
            character fname*12,fnam*12
            write(6,200)
200         format('enter the input signal levels file name')
100         format(a12)
            read(5,100)fname
            open(unit=11,file=fname,status='old')
            write(6,301)
301         format(' enter the output file name ')
            read(5,100)fnam
            inquire(file=fnam,exist=used)
            if(.not.used) goto 700
            open(unit=12,file=fnam,status="old")
            goto 701
700         open(unit=12,file=fnam,status='new')
701         do 5 i=1,100
            x(i)=-130.0+0.5
            ii=0
            j=0
3           read(11,777,end=4) buf
            if(buf.le.x(i)) j=j+1
            ii=ii+1
            goto 3
4           rewind(11)
            z(i)=float(j)/float(ii)
            if (i.eq.1) y(1)=0
            if (i.eq.1)goto 133
            y(i)=(z(i)-z(i-1))/(x(i)-x(i-1))
133         write(12,7)x(i),y(i)
            write(6,7)x(i),y(i),z(i)
5           continue
6           format(i16)
777         format(f20.10)
7           format(g20.6,1x,g20.6,1x,g20.6)
            step=1.
            del=1.
            k=0
110         sum=0.0
            do 50 i=1,100
            xx=2.*dsqrt(dabs(x(i))*(1.+step)*step)
            pp=fun(xx)*(1.+step)*dexp(-(x(i))*(1.+step)-step)
50          sum=sum+(y(i)-pp)**2
            k=k+1
           if(k.eq.1)goto 80
            if(sum.ge.sum1)goto 120
```

```
80          step=step+del
            sum1=sum
            goto 110
120         continue
            write(6,300)step
300         format(' the K factor = ',g20.6)
            do 121 i=1,100
            xx=2.*dsqrt(dabs(x(i))*(1.+step)*step)
            pp=fun(xx)*(1.+step)*dexp(-(x(i))*(1.+step)-step)
            write(6,18)x(i),pp
18          format(' x  =  ',g20.6,2x,'pp  =  ',g20.6)
121         write(12,7)x(i),pp
            stop
            end
C  MODIFIED BESSEL FUNCTION OF ORDER ZERO.
            double precision function fun(xx)
            double precision fun,z,r,xx
            r=dabs(xx)
            if(r-3.75d0)1,1,2
1           z=xx*xx*7.111111d-2
            fun=(((((4.5813d-3*z+3.60768d-2)*z+2.659732d-1)*z
            +1.206749d0)*z+3.089942d0)*z+3.515623d0)*z+1.d0
            return
2           z=3.75d0/r
            fun=dexp(r)/dsqrt(r)*((((((((3.92377d-3*z
        c   -1.647633d-2)*z+2.635537d-2)*z-2.057706d-2)*z
        c   +9.16281d-3)*z-1.57565d-3)*z+2.25319d-3)*z
        c   +1.328592d-2)*z+3.989423d-1)
            return
            end
```

Symbolic File PIFEX12.F

```
      c this program computes frequaency of fade duration
            dimension ix(100),iy(100)
            logical used
            character fname*12,fnam*12
            write(6,200)
200         format(' enter the input signal levels
            file name ')
            read(5,100)fname
            write(6,301)
301         format(' enter the output file name')
            read(5,100)fnam
100         format(a12)
            write(6,302)
302         format(' enter the desired signal level ')
            read(5,303)sig
303         format(g20.10)
            open(unit=11,file=fname,status='old')
            inquire(file=fnam,exist=used)
            if(.not.used)goto 700
            goto 701
700         open(unit=12,file=fnam,status='new')
701         read(11,777)buf
            ii=0
            do 400 i=1,100
            iy(i)=0
400         ix(i)=i
99          read(11,6,end=98)buf
            ii=ii+1
            goto 99
98          ii=ii+1
            write(6,7)ii
            j=0
            rewind(11)
3           read(11,6,end=4) buf
            fi=buf
            if(fi.le.sig) then
            j=j+1
            goto 3
            end if
            if(fi.gt.sig.and.j.ne.0) then
            jbin=j/2
            if(j.eq.1)then jbin=1
            if(jbin.gt.100)jbin=100
            iy(jbin)=iy(jbin)+1
            j=0
            end if
            goto 3
4           do 211 i=1,100
            write(12,7)ix(i),iy(i)
211         write(6,7)ix(i),iy(i)
6           format(i16)
```

```
777         format(f20.10)
7           format(i16,10x,i16)
            stop
            end
```

Symbolic File PIFEX13.F

```
c this program generates the downloadable file out of signal
c levels file and the signal phase files for the use of the
c AKPLOT routine to produce the plots.
            dimension x(1000),y(1000)
            logical used
            character fname*12,fnam*12
            write(6,200)
200         format(' enter the input file name ')
            read(5,100)fname
            write(6,333)
333         format("  ")
            write(6,301)
301         format(' enter the output file name')
            read(5,100)fnam
100         format(a12)
            write(6,333)
            open(unit=11,file=fname,status='old')
            inquire(file=fnam,exist=used)
            if(.not.used) goto 700
            open(unit=11,file=fnam,status="old")
            goto 701
700         open(unit=12,file=fnam,status='new')
701         do 5 i=1,1000
            x(i)=0.001*i
            read(11,777)buf
            y(i)=buf
            write(12,7)x(i),y(i)
            write(6,7)x(i),y(i)
5           continue
6           format(i20)
777         format(f20.10)
7           format(f20.10,1x,f20.10)
            close(11)
            close(12)
            stop
            end
```

Symbolic File PIFEX14.F

```
c This program computes the fourier transform of 1024
c samples of the signal levels file and the output is stored
c into the output file supplied by the user.
           dimension x(2,2100),yy(520)
           logical used
           character fname*12,fnam*12
           write(6,200)
200        format(' enter the input file name ')
           read(5,100)fname
           write(6,301)
301        format(' enter the output file name')
           read(5,100)fnam
100        format(a12)
           open(unit=11,file=fname,status='old')
           inquire(file=fnam,exist=used)
           if(.not.used) goto 700
           open(unit=12,file=fnam,status="old")
           goto 701
700        open(unit=12,file=fnam,status='new')
701        do 4 i=1,1024
           read(11,777,end=4) buf
           x(1,i)=buf
           x(2,i)=0.0
4          continue
           sign=-1.0
           npow=9
           t=0.001
           call fftran(sign,t,x,npow)
           do 11 i=1,512
           yy(i)=sqrt(x(1,i)**2+x(2,i)**2)
11         continue
           do 12 i=1,999
           nc=i-500
           xxx=float(nc)/1.024
           if(nc.eq.-1.or.nc.eq.0)goto 12
           if(nc.eq.1) xxx=0.0
           yyy=yy(abs(nc))
           write(12,7)xxx,yyy
           write(6,7)xxx,yyy
12         continue
6          format(i16)
777        format(f20.10)
7          format(f20.10,1x,f20.10)
           stop
           end
           DIMENSION X(1),CS(2),MSK(13)
           COMPLEX  X,CXCS,HOLD,XA
           EQUIVALENCE (CXCS,CS)
           NMAX=2**NPOW
           ZZ=6.283185306*SIGN/FLOAT(NMAX)
           DELTA=T
```

```
      IF(SIGN) 10,10,5
5     DELTA=1./(T*FLOAT(NMAX))
10    MSK(1)=NMAX/2
      DO 15 I=2,NPOW
15    MSK(I)=MSK(I-1)/2
      NN=NMAX
      MM=2
      DO 45 LAYER=1,NPOW
      NN=NN/2
      NW=0
      DO 40 I=1,MM,2
      II=NN*I
      W=FLOAT(NW)*ZZ
      CS(1)=COS(W)
      CS(2)=SIN(W)
      DO 20 J=1,NN
      II=II+1
      IJ=II-NN
      XA=CXCS*X(II)
      X(II)=X(IJ)-XA
20    X(IJ)=X(IJ)+XA
      DO 25 LOC=2,NPOW
      LL=NW-MSK(LOC)
      IF(LL) 30,35,25
25    NW=LL
30    NW=MSK(LOC)+NW
      GO TO 40
35    NW=MSK(LOC+1)
40    CONTINUE
45    MM=MM*2
      NW=0
      DO 80 I=1,NMAX
      NW1=NW+1
      HOLD=X(NW1)
      IF(NW1-I) 60,55,50
50    X(NW1)=X(I)*DELTA
55    X(I)=HOLD*DELTA
60    DO 65 LOC=1,NPOW
      LL=NW-MSK(LOC)
      IF(LL) 70,75,65
65    NW=LL
70    NW=MSK(LOC)+NW
      GO TO 80
75    NW=MSK(LOC+1)
80    CONTINUE
      RETURN
      END
```